DESCRIPTION OF PROGRESS

Investigations of several subproblems in the area of derivation of parallel programs were continued during the current quarter. These investigations include:

(1) Michael Landis (graduate student), John Reif (PI), and Robert Wagner (Duke faculty): Intermediate Representation for Parallel Implementation

Our research efforts are just beginning to focus on the possibility of extending a high-level data-parallel language with constructs for process parallelism. Our goal is to begin with a data-parallel language like NESL, which is under development by Guy Blelloch at Carnegie Mellon University. This language provides nested data-parallelism. We believe that by extending it with process parallel primitives, the language will have wider applicability, but yet will still be able to be implemented efficiently.

This work has evolved directly out of research over the past year in trying to develop extensions to a low-level data-parallel intermediate representation to accommodate asynchronous processes. After extensive research we have decided that parallel process extensions to a low-level intermediate representation are not practical because of fundamental differences in primitives provided by different hardware vendors. Instead, we are focusing our efforts currently on the extension of a run-time library for implementing data-parallel languages. This library will provide the support for high-level language development while maintaining portability and efficiency through the use of the C language.

As an example, one possibility which we are investigating is the integration of the POSIX thread package with CVL, the C Vector Library *under development at CMU.* In order to gain experience with parallel systems and with the implementation of CVL, Mike Landis is collaborating with Blelloch's team in order to develop a multiprocessor CRAY implementation of CVL. This work should be done around the end of October, at which time we will focus on the extensions to this library.

(2) Michael Landis (graduate student), John Reif (PI), and Robert Wagner (Duke faculty): Data Movement on Processor Arrays

We are still developing ways of evaluating uniform expressions in near minimum parallel time on processor arrays. A paper describing the solution on two-dimensional arrays as been submitted for journal publication; an abstract of this paper follows.

"Evaluating Uniform Expressions Within Two Steps
of Minimum Parallel Time",
Robert A. Wagner

ABSTRACT

Consider an array of Processing Elements [PEs], connected by a 2-dimensional grid network, and holding at most one operand of an expression in each PE. Suppose that each PE is allowed, in any one parallel step, to receive one item of data from any of its 4 immediate neighbors, and to transmit one datum, as well. How can an associative operator, such as addition, combine all the operands, using as little time for communication as possible? An expression using such a single operator is termed a uniform expression. When the total number of communication links used is the measure of goodness, this problem becomes a Steiner Tree problem, in the Manhattan Distance metric. When the measure is *minimizing* the parallel time to completion, a method for solving this problem is given which is optimal to within an additive constant of 2 time-steps. The method has applications when the operands are matrices, spread over an array of PEUs, as well. Some lower bounds for this problem, in more general networks, are also proven.

Current work

Our current work in this area is to extend this parallel reduction operation to higher dimensional grids. This work is nearing its successful completion. Robert Wagner and Mike Landis have developed a method for performing reductions on multidimensional processor arrays that is within a few steps of a provably minimal time. They are currently finishing their paper, which is a follow-up to Robert Wagner's paper, "Evaluating Uniform Expressions Within Two Steps of Minimum Parallel Time." This paper solved the problem for two-dimensional arrays only.

Future work

In looking at the problems of distributing collection-oriented operations and collections across many MIMD processors, the question of data-communication cost comes up.

Suppose PEUs are connected in a 2-D grid, with the property that any PE can receive a datum from any one neighbor at a given time-step. (Other network models, and PE communication behavior schemes are also possible.) In this setting, ignoring operation costs, we've studied the problem of computing >_ vi , where each vi is originally located on a *different PE. The goal is to minimize parallel communication time.* We have written a TR that solves the problem within 2 steps of optimality regardless of the initial placement of the operands in the grid.

This sort of study opens an area of research, delimited by choosing different combinations of assumptions about PE behavior, network topology, and problems to be solved. A systematic study of these questions, oriented toward problems which are communication-intensive and of interest to people developing packages like LINPAK, seems in order.

(3) Peter Mills (Research Associate) with John Reif: Parallel Models with Tagged-Memory and Rate Control

Summary:
We are extending high-level parallel computation models with abstractions for asynchronous tagged memory communication and constraints on relative rates of progress, and are conducting preliminary investigations into transformation techniques to efficiently realize these abstractions on practical parallel machines. We have introduced novel concurrency constructs which mimic tagged memory into a high-level parallel language Proteus, and are investigating transformation techniques targeting lower-level languages such as the C Vector Library (CMU). At the same time we are investigating extending an existing widely portable data-parallel language, CMU's NESL (supporting nested data parallelism) with a wrapper for asynchronous parallelism built on shared state and result-parallelism (MultiLisp futures). The intent is to extend and thus capitalize on existing techniques for transforming *nested data parallelism to vector models. The introduction of rate control* supports a succinct specification of intended resource allocation, and is a first step in extending models of parallel computation with real-time properties, such as processor rates, in order to support timing analysis. Current models, variants of the PRAM such as the APRAM and HPRAM, only accommodate asynchrony of control or hierarchy of control and communication.

Details:
We have developed a new parallel programming construct, the rate construct, which constrains relative rates of progress of parallel processes.

The rate construct specifies constraints on the relative rates of progress of tasks executing in parallel, where progress is the amount of computational work as measured by elapsed ticks on a local virtual clock. By prescribing expected work, rates abstractly specify the allocation of processors to tasks needed to achieve that work, effected for example by load balancing. We have developed a simple notion, fixed-rate, which apportions an unvarying percentage of processor time to each process with no assumptions of global clock synchronization between processors. The utility of the rate construct has been examined for a variety of problems, including weighted parallel search for a goal, adaptive many-body simulation in which rates abstract the requirements for load-balancing, and variable time-stepped computations in which the use of rates can alter the frequency of asynchronous iterations. We are currently investigating means of transforming rate primitives to lower-level real-time and scheduling constructs.

We are integrating the tagged-memory model into the parallel language, Proteus, to serve as a vehicle for refinement techniques. We have introduced a variant of "synchronization variables" and a novel construct we call "linear variables". Synchronization variables, common to coordination languages such as PCN and CC++, are a synchronization mechanism in which processes must wait for an "empty" variable to become defined when its value is required in an evaluation. Linear variables are a further extension which model resource consumption, and prove valuable in succinctly modeling channel and rendezvous operations within a shared-memory framework. We are extending this model with other shared-memory abstractions for specifying process topology and directing non-local references to processes. Synchronization variables, in combination with other features such as barriers for expressing loosely synchronous computations (SPMD and SIMD), prove particularly advantageous in that they can be used to map directly to tagged-memory machines such as the J-machine, or serve as a foundation for implementing primitives on other machines.

Ongoing work:
Development of an MSIMD language extending CMU's nested data-parallel language NESL with a wrapper for asynchronous process execution based on shared variables and MultiLisp futures.

Development of refinement techniques for transforming extended NESL to threads of vector code, targeting such machines as Cray YMP and CM-5.

Demonstration of viability of these techniques through concrete implementations of N-body algorithms, specifically clustering and Fast Multipole Methods, targeting MPSMD machines (e.g., CM-5).

(4) Peter Su (postdoc) and John Reif: Implementations of Parallel Algorithms in Computational Geometry

We have been working on the implementational aspects of parallel algorithms. Specifically, we have been studying parallel algorithms for constructing Voronoi Diagrams and related problems. Our interest in this study is not only to build effective algorithms for these problems, but also to consider the kinds of tools that make such work easier and more effective.

Our work has been broken up into three stages:

(1) Study the theory and practice of conventional algorithms for this problem.
(2) Study the current body of theoretical work on parallel algorithms for this problem.
(3) Using the knowledge gained in (a) and (b), design and implement parallel algorithms for this problem on several machines. Then study the performance of these algorithm and how well the theoretical results match the behavior of the implementation.

We have been actively working on stages (1) and (2) for the last few months and we are now ready to move on to stage (3). The study of practical sequential algorithms has been especially helpful in the pursuit of simple and efficient parallel algorithms, since they provide a good set of ideas to extend and refine in a parallel setting.

Using the experience that we gain from this work, we are also investigating and planning tools that could aid the programmer in implementing effective parallel algorithms. Since many parallel algorithms, especially in computational geometry, have similar structure, one could imagine a tool for reasoning about abstract classes of algorithms. In particular, such a system could aid the programmer in tuning performance parameters for specific machines based on architectural characteristics such as global memory bandwidth and latency, processor speed, local memory size, and so on. Also, more basic tools for doing visualization and performance analysis are needed to help the programmer to effective experimental analysis of his implementations. Tools for profiling, animation, simulation and data

analysis would all be extremely useful in these settings. At this point, there are no such tools widely available to the research community.

Initial implementations of the ideas in this work has begun and has been successful. I presented a paper at the DAGS conference this summer that describes Cray algorithms for basic proximity problems. I have also begun to explore implementations of the other ideas on various machines, including the MasPar MP-1, the CM-5, and the KSR-1. This development work will make up a large part of my PhD thesis, which should be finished by this spring.

In addition, we have designed an efficient algorithm for constructing Delaunay triangulations which we are in the process of implementing on the KSR-1. It uses a novel 'transactional' method of constructing the diagram in incremental phases. Each phase attempts to add as many points as possible in parallel, but if two insertions conflict, then one or the other must back off. We structure the insertion phases in a way that is reminiscent of transaction processing systems so that the current diagram is guaranteed to be unique. In addition, we randomize the insertion order of the points to guarantee that the algorithm can achieve sufficient parallelism.

(5) Shenfeng Chen with John Reif: Parallel Sort Implementation

The fastest known sort is a parallel implementation of radix sort in a CRAY, due to CMU's Guy Blelloch. The current sorting algorithms on parallel machines like Cray and CM-2 use radix and bucket sort. But they are not taking advantage of possible distribution of the input keys. We are developing an algorithm using data compression to achieve a fast parallel algorithm which takes this advantage. We expect the new algorithm to beat the previous fastest sort by a few factors. We are working to implement this new parallel sorting algorithm on various parallel machines.

Details:
Radix sort is very efficient when the input keys can be viewed as bits. But the basic radix sort is not distribution based so it needs to look up all digits.

Our approach is to find the structure (distribution) of the input. This is achieved by sampling from the original set. Then a hash table is build from those sample keys. All keys are indexed to buckets separated by

consecutive sample keys. A probability analysis shows that the largest set can be bounded within a constant of the average size.

The indexing step is made faster by binary searching the hash table for match. From previous result, each hash function computation needs only constant time.

Our algorithm needs O(nloglogn) time in sequential given that the compression ratio of the given input set is not too big. In parallel, our algorithm works well in chain-sorting. In list ranking sorting, the total work is also reduced.

We have implemented this algorithm on Sparc II and compared its performance with the system routine quicksort. It turns out that our algorithm outwins the quicksort() for sufficiently large number of keys (32M). Thus, it may find its place in sorting large database operations (e.g., required by joint operations). In these applications the keys are many words long so our algorithm is even more advantageous in this case where the cutoff is much lower.

(6) Deganit Armon (A.B.D.) with John Reif: Dynamic Graph Separator Algorithms.

Summary:
We worked on the problem of dynamically maintaining graph separators when the input graph is changed by adding or deleting vertices and edges. Using a randomized algorithm, we are able to dynamically maintain separators for a changing graph in polylog time. Our dynamic algorithms are based on static separator algorithms developed recently by Miller et. al. for a large class of geometrically defined graphs, called overlap graphs, which includes planar graphs. Our results can be applied to generate dynamic algorithms for a wide variety of combinatorial and numerical problems, whose underlying graph is a dynamically changing overlap graph.

Details:
Graph separators are important in the context of designing divide and conquer algorithms for graph problems. A graph separator is a small subset of vertices of the graph which, when removed, partition the graph into unconnected subgraphs. A separator is good if it is small and achieves a partition into subgraphs of roughly equal size. A separator tree for a

graph is a recursive structure that stores separators for the graph and its separated subgraphs.

Given a graph G from a large class of graphs called overlap graphs (a superset of planar graphs), Teng showed a linear time, randomized algorithm for finding a good separator for G. We show that if G is dynamically changing by that addition and deletion of vertices, it is possible to maintain the separator in expected time O(log n). Furthermore, we show that a separator tree for G can be maintained in $O(\log^3 n)$. This is the first known polylog dynamic algorithm for maintaining separators of a graph.

We also gave a general technique for transforming expected time randomized algorithms to high likelihood time randomized algorithms. Using this technique, it is still possible to maintain separators for an overlap graph in polylogarithmic time. We show that separators can be maintained with high likelihood in $O(\log^3 n)$ time.

These algorithms can be applied to dynamic problems that have an underlying graph structure, such as path problems in dynamically changing graphs, or dynamic nested dissection for solving linear systems in which matrix values and the sparsity structure of the matrix are changing.

(7) Prokash Sinha with John Reif: Randomized Parallel Algorithms for Min Cost Paths

Summary:
We have completed our initial investigation to derive randomized parallel algorithms for Min Cost Paths in a Graph of High Diameter. Our present accomplishment is a randomized sequential algorithm with an order of magnitude performance gain for some dense graphs.

We also found a similar result for PRAM computational model which meets the work we proposed to do in our paper "A Randomized Algorithm for Min Cost Paths in a Graph of High Diameter: Extended Abstract" (J. Reif and P. Sinha). Currently we are in the process of submitting our findings to technical journals and conferences. Our next phase of work would include similar derivations of randomized parallel algorithms for a wide variety of discrete structures which arises naturally in the area of Graph Theory and Combinatorics. Our current research effort is to extend the techniques of Flajolet and Karp to develop techniques and tools for timing analysis of

algorithms. This effort is to derive tools for semiautomatic randomized analysis.

(8) Hongyan Wang with John Reif: Social Potential Fields: A Molecular Dynamics Approach for Distributed Control of Multiple Robots.

Summary:
Much of the early research in robotic planning and control has considered the case of only a single robot. There is now a number of robot systems which include a small number of autonomous robots and consequently there is a quickly growing literature on the planning and cooperative control of systems of small numbers of robots. Our work is concerned with Very Large Scale Robotic (VLSR) systems consisting of at least hundreds to perhaps tens of thousands or more autonomous robots. Our molecular dynamics approach is distributed and robust and flexible.

Details:
We view our VLSR systems as a molecular dynamics system, with predefined force laws between each ordered pair of components (robots, obstacles, objectives and other configurations). These force laws are similar to those found in molecular dynamics, incorporating both attraction and repulsion in the form of inverse power laws. However these laws may differ from molecular systems in that we allow the controller to arbitrarily define distinct laws of attraction and repulsion for separate pairs and groups of robots to reflect their social relations or to achieve some goals. For example, we define a pair-wise force law of attraction and repulsion for a group of identical robots. The repulsion will prevent collision among robots and the attraction will keep them in a cluster. This simulates the phenomena called "individual distance" in sociobiology.

Once the force laws are set up (they can be modified by the global controller), each individual's movement is computed locally according to the local environment sensed by individual robots and the force laws. Thus the control is distributed and robust. Each robots obeys Newton's Law and makes movement complying to the total force on it from the other components.

We give concrete examples to show that this distributed autonomous control will have lots of applications in industry, military and other areas in the future when costs for individual robots drop and robots can be made much more compact and more capable and flexible.

We did computer simulations involving large numbers of robots. Some interesting and useful patterns can be achieved by defining proper force laws for the system, e.g. forming a more or less evenly distributed single cluster, forming a circle to guard a static point particle standing for castle. We are doing more simulations showing more complex patterns.

We also discuss about spring laws similar to molecular bondings to robotic control. Theories of graph rigidity support that we can design a VLSR system which has a rigid structure. This has also applications where assemblies are needed to finish some job efficiently.

(9) Hongyan Wang with John Reif: A Constant Time Algorithm for N-body Simulation with Smooth Distributions.

Summary:
N-body simulation problem is as follows: Given N points that have pair-wise interactions, compute the equilibrium configuration of the N points. This problem is central to a large body of work in theoretical physics, chemistry, and scientific computing, including: cosmology, plasma simulation, molecular dynamics, and fluid mechanics. The fastest N-body simulation algorithm due to Greengard has time complexity of O(N) for one step simulation. We propose to use the concept of density function to describe the configuration of the large particle system and a method to compute the equilibrium density function iteratively when given the initial density function in constant time with the time complexity depending only on the potential function and the required precision.

Details:
In a system of large number, say millions of particles, we are interested more in the structure of the system, especially the structure under equilibrium conditions than in the exact positions of all particles. Observations from many fields, such as cosmology, plasma simulations, molecular dynamics, and fluid mechanics, suggest that the distribution of particles is homogeneous and can be described by smooth functions. Thus we propose to use density function to describe the configuration of particle systems.

Based on the fact that under equilibrium conditions, the total force on each particle should equal to 0, we derive an iterative procedure IMPROVE for improving the density function, which is of the form $\Phi^{(n+1)}(x)=\text{IMPROVE}(\Phi^{n}(x))$, where x is a position in the domain of interest. Computing the total force on one robot by summing up

discretely all the forces from other robots will require Omega(n) time. Instead, we only sum up forces from a constant number of nearby particles. For particles far away, we do an integration of force function multiplied by density function to approximate the resultant force. This reduces the time complexity to constant. Thus each improvement procedure requires constant time and the number of iterations depends on the required precision, and thus can be constant.

Simulations showed that the iterative improvement procedures converges. The results showed that in 1-d the density function has a bell-shaped curve and in 2-d has a vault-shaped surface in the domain of interest and outside the domain has 0 value.

(10) Akitoshi Yoshida with John Reif: Image and Video Compression

We considered several compression techniques using optical systems. Optics can offer an alternative approach to overcome the limitations of current compression schemes. We gave a simple optical system for the cosine transform. We designed a new optical vector quantizer system using holographic associative matching and discussed the issues concerning the system.

Optical computing has recently become a very active research field. The advantage of optics is its capability of providing highly parallel operations in a three dimensional space. Image compression suffers from large computational requirements. We propose optical architectures to execute various image compression techniques, utilizing the inherent massive parallelism of optics.

In our paper[RY2], we optically implemented the following compression and corresponding decompression techniques:

- o transform coding
- o vector quantization
- o interframe coding for video

We showed many generally used transform coding methods, for example, the cosine transform, can be implemented by a simple optical system. The transform coding can be carried out in constant time.

Most of this paper is concerned with an innovative optical system for vector quantization using holographic associative matching. Limitations of conventional vector quantization schemes are caused by a large number of

sequential searches through a large vector space. Holographic associative matching provided by multiple exposure holograms can offer advantageous techniques for vector quantization based compression schemes. Photo-refractive crystals, which provide high density recording in real time, are used as our holographic media. The reconstruction alphabet can be dynamically constructed through training or stored in the photorefractive crystal in advance. Encoding a new vector can be carried out by holographic associative matching in constant time.

We also discussed an extension of this optical system to interframe coding.

On going work:
We are investigating optical algorithms for video compression.

(1) Computational Geometry by Optical Computers
Some problems require inherently high degrees of interconnections which may not be provided by any conventional electrical computers. The advantage of optical computers is their apparent parallelism in a three dimensional space. Several computational models have been already proposed and constructed by various research groups. As the progress of optical computers continues, there is a great demand in designing and investigating various algorithms that are efficient and appropriate for the proposed models. This situation resembles to the one a decade ago, when various algorithms were investigated for the theoretical VLSI model. Thus, we understand that the investigation on optical computing algorithms will be essential to the development of optical or hybrid massively parallel computers.

Optical techniques are particularly suited for processing images. This leads us to believe that many problems found in computational geometry may be efficiently solved by optical computers. Some researchers have recently started to investigate some basic problems. We have been investigating these and some other problems. We have obtained some new results.

(2) Optical Interconnection
Among processing units placed on a plane, various space-invariant interconnections can be holographically established in constant time. We are investigating appropriate interconnections and efficient algorithms for several problems.

(3) Efficient computation for optical scattering
An efficient algorithm to solve the Helmholtz equations was developed by Rokhlin at Yale. We have been studying his algorithm.

(4) Simulation of optical computing algorithms
We implemented a software simulator for optical computing algorithms. The simulator is written in C on the X-window environment. It has a lisp-like user interface, and images, which are the basic data structures in the optical computing algorithms, are treated as lisp objects. We simulated some algorithms designed for computational geometry problems.

We are improving the simulator and planning to implement it on a parallel machine.

(11) Researchers supported (other than PI):

Salman Azhar, graduate student
Mike Landis, graduate student
Peter Mills, post-doc
Peter Su, visiting graduate student
Robert Wagner, professor
Akitoshi Yoshida, graduate student

(13) Papers

(1) Algebraic Methods for Testing the k-Vertex Connectivity of Directed Graphs, (with J. Cheriyan), *3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, 1992. Accepted for publication as "Directed s-t Numberings, Rubber Bands, and Testing Digraph k-Vertex Connectivity," in *Combinatorica*.

(2) Searching in an Unknown Environment (with M. Kao and S. Tate), *4th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA92), San Diego, CA, 1992.

(3) Linear Time Approximate Evaluation of a Polynomial at Real Points (with V. Pan). *32rd Annual IEEE Symposium on Foundations of Computer Science* (FOCS92), Pittsburgh, PA., Oct 1992.

(4) The Power of Combining the Techniques of Algebraic and Numerical Computing: Improved Approximate Multipoint Polynomial Evaluation and Improved Multipole Algorithms (with V.Y. Pan and S.R. Tate), *33rd Symposium on Foundations of Computer Science*, October 1992. Also submitted for journal publication as "The Complexity of Trummer's Problem, Zeta Function Evaluation, and N-body Simulation" (with S. Tate).

(5) Expected Parallel Time and Sequential Space Complexity of Graph and Digraph Problems (with P. Spirakis). *Algorithmica*, Vol. 7, pp. 597-630, 1992.

(6) Fast and Efficient Parallel Solution of Sparse Linear Systems (with V. Pan). Accepted for publication in *SIAM Journal on Computing*, 1992.

(7) Nested Annealing: A Provable Improvement to Simulated Annealing (with S. Rajasekaran). Accepted for publication in *Journal of Theoretical Computer Science*, November 1992.

(8) On Threshold Circuits and Efficient, Constant Depth Polynomial Computation (with S. Tate). Accepted for publication in *SIAM Journal of Computing*, 1992.

(9) Planarity Testing in Parallel (with V. Ramachandran), University of Texas at Austin Technical Report TR-90-15, June 1990. Invited to special issue of *Journal of Algorithms*, 1992.

(10) The Computability and Complexity of Ray Tracing (with D. Tygar and A. Yoshida). To appear in *Discrete & Computational Geometry*, 1992.

(11) Randomized Algorithms for Binary Search and Load Balancing on Fixed Connection Networks with Geometric Applications (with S. Sen). *2nd Annual ACM Symposium on Parallel Algorithms and Architectures*, Crete, Greece, July 1990, pp. 327-337. To appear in *SIAM Journal of Computing*, 1992.

(12) Strong k-connectivity in Digraphs and Random Digraphs (with P. Spirakis). Accepted for publication in *Algorithmica*, 1992.

(13) Probabilistic Parallel Prefix Computation. Accepted for publication in *Computers and Mathematics with Applications*, 1992.

(14) Efficient VLSI Fault Simulation. To appear in *Computers and Mathematics with Applications*, 1992.

(15) Continuous Alternation (with S. Tate). To appear in a special issue of *Algorithmica*, edited by B. Donald, 1992.

(16) Quad Tree Structures for Image Compression Applications (with T. Markas). Special issue of *Journal of Information Processing and Management*, 1992.

(16) Memory-Shared Parallel Architectures for Vector Quantization Algorithms (with T. Markas), 1992. Accepted for the Picture Coding Symposium, Lusanne Switzerland, Mar 93. Submitted for journal publication.

(17)A Method for Deriving Systolic Algorithms (by R.A. Wagner and M.D. Landis), 1992. Submitted for journal publication.

(18) Evaluating Uniform Expressions Within Two Steps of Minimum Parallel Time (by R.A. Wagner), 1992. Submitted for journal publication.

(19)Shortest Paths in Euclidean Space with Polyhedral Obstacles (with J.A. Storer). *Symposium on Mathematical Foundations of Computer Science*, Czechoslovakia, August 1988. Revised as "Shortest Paths in the plane with polygonal obstacles", 1992. Submitted for journal publication.

(20) Optical Expanders with Applications in Optical Computing (with A. Yoshida), 1992. Submitted for journal publication.

(21) On Applications of Crypto-complexity to Analyzing Efficiency of Capital Markets (with S. Azhar), 1992. Submitted for journal publication.

(22) Fully Dynamic Graph Connectivity in Logarithmic Expected Time (with P. Spirakis and M. Yung), 1992. Submitted for journal publication.

(23) On Parallel Implementations and Experimentations of Lossless Data Compression Algorithms (with T. Markas), 1992. Submitted for publication.

(24) A Randomized Algorithm for Min Cost Paths in a Graph of High Diameter: Extended Abstract (with P. Sinha), 1992. Submitted for publication.

(25) Fast Algorithms for Closest Point Problems: Practice and Theory (by Peter Su), 1992. Submitted for publication.

(26) A Fast Sort and Priority Queue for Entropy Bounded Inputs (with Shenfeng Chen), 1992. Submitted for publication.

(27) Rate Control as a Language Construct for Parallel and Distributed Programming (with Peter Mills), 1992. Submitted for publication.

(28) Social Potential Fields: A Molecular Dynamics Approach for Distributed Control of Social Behavior in Robots (with Hongyan Wang), 1992.

(29) Dynamic Algebraic Algorithms (with S. R. Tate), 1992. Submitted for publication.

(30) Dynamic Parallel Tree Contraction (with S. R. Tate), 1992. Submitted for publication.

(31) A Dynamic Separator Algorithm with Applications to Computational Geometry and Nested Dissection" (with D. Armon), 1992. Submitted for publication.

(32) Using Learning and Difficulty of Prediction to Decrease Computation: A Fast Sort and Priority Queue on Entropy Bounded Inputs (with S. Chen), 1992. Submitted for publication.

(33) Parallel and Output Sensitive Algorithms for Combinatorial and Linear Algebra Problems (with J. Cheriyan), 1992. Submitted for publication.

(34) Re-Randomization and Average Case Analysis of Fully Dynamic Graph Algorithms (with P.G. Spirakis and M. Yung), 1992. Submitted for publication.

(35) The Complexity of N-body Simulation (with S.R. Tate), 1992. Submitted for publication.

(36) Strictly Polylog Time, Linear Space Algorithms for Nearest Neighbor Search and Dynamic Separators in d-Dimensions (with D. Armon) 1992. Submitted for publication.

(37) Multispectral Image Compression Algorithms (with A. Markas), 1992. Submitted for publication.